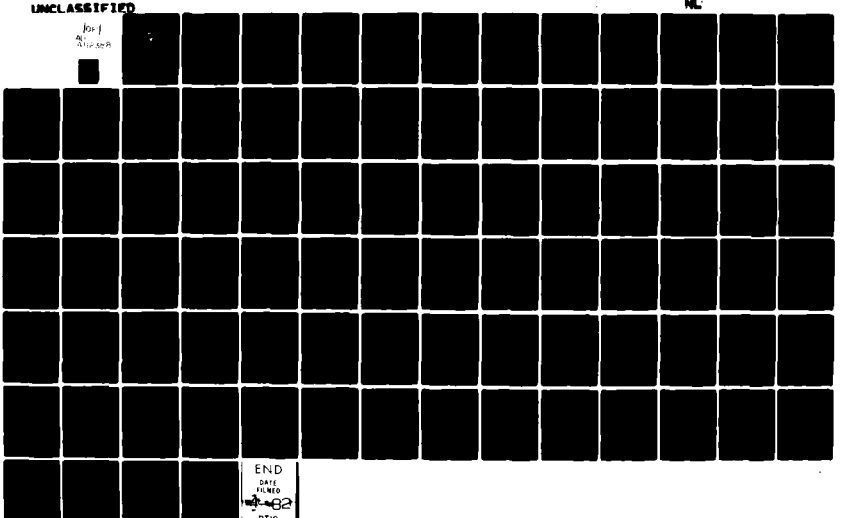
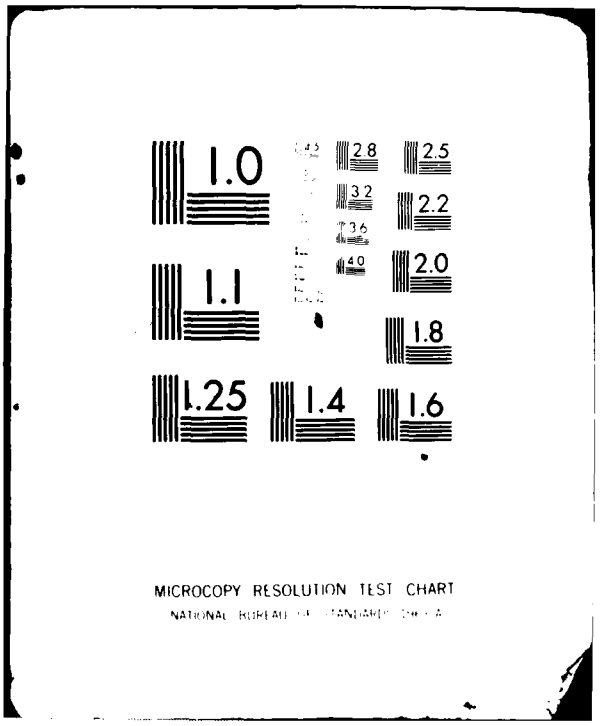
AD-A112 308 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 9/2
ADAPTATION OF A KNOWLEDGE-BASED DECISION-SUPPORT SYSTEM IN THE --ETC(U)
DEC 81 W C CLAIR, R F DANHOF

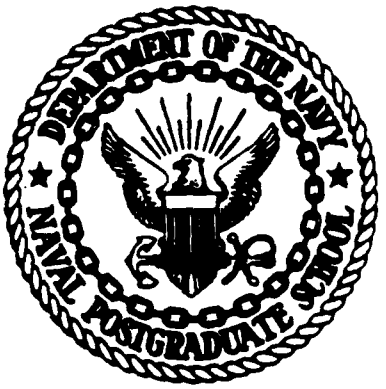
UNCLASSIFIED NL



END
DATE FILMED
4-82
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART

NAVAL POSTGRADUATE SCHOOL
Monterey, California



THESIS

ADAPTATION OF A KNOWLEDGE-BASED
DECISION-SUPPORT SYSTEM
IN THE TACTICAL ENVIRONMENT

by

William C. Clair

and

Ronald F. Danhof

December 1981

Thesis Advisor: D. R. Smith

Approved for public release; distribution unlimited

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A112 388 / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Adaptation of a Knowledge-based Decision-support System in the Tactical Environment	5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; December 1981
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) William C. Clair Ronald F. Danhof	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940	12. REPORT DATE December 1981 13. NUMBER OF PAGES 85
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Artificial Intelligence; Decision-Support Systems; Tactical Decision-making; Knowledge-based Decision-support System; Adaptable Intelligent Systems

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A knowledge-based computer system, with its foundation in Artificial Intelligence, would be a valuable asset to the military tactical commander. Current systems are slow, large, expensive, inflexible and therefore, impractical for use in the tactical environment. A detailed design of a prototype small-computer-based system is presented which processes and interprets intelligence and tactical information to assist tactical commanders in

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

making decisons. The system, TAC*, for "Tactical Adaptable Consultant", incorporates a database, and a distributed interface. Emphasis is placed on the representation and processing of two types of information: data about the real world; and knowledge about what that data means.

Approved for public release; distribution unlimited

Adaptation of a Knowledge-based Decision-support System in the Tactical Environment

by

William C. Clair
Lieutenant Commander, United States Navy
B.S., United States Naval Academy, 1973

Ronald F. Danhof
Captain, United States Army
B.S., United States Military Academy, 1973

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 1981

Authors: William C. Clair

Ronald F. Danhof

Approved by: D. C. R. Smith
Thesis Advisor

Second Reader

Chairman, Department of Computer Sciences

W. M. Woodyard
Dean of Information and Policy Sciences

ABSTRACT

A knowledge-based computer system, with its foundation in Artificial Intelligence, would be a valuable asset to the military tactical commander. Current systems are slow, large, expensive, inflexible and therefore, impractical for use in the tactical environment. A detailed design of a prototype small-computer-based system is presented which processes and interprets intelligence and tactical information to assist tactical commanders in making decisions. The system, TAC*, for "Tactical Adaptable Consultant," incorporates a database, a knowledge base, their associated management systems, and a distributed interface. Emphasis is placed on the representation and processing of two types of information: data about the real world; and knowledge about what that data means.

TABLE OF CONTENTS

I. INTRODUCTION AND BACKGROUND

A. INTRODUCTION

1. Focus

Although computers are being used for many applications in the military, the area of tactical decision-support is of the most critical importance. This area combines the many aspects of typical business decision-support with an urgency which is only found in the practice of war. The Tactical Commander must select a course of action which will result in the attainment of certain goals (e.g. the destruction of enemy forces or the survival of one's own forces).

It is the potential for the loss of many human lives and, more importantly, the consequences of failure in the political-military effort which makes this problem so critical. Because of its crucial nature, tactical decision-support is the area where we will focus our design.

2. The Problem

Simply stated, the problem is to provide the tactical commander with decision-support which is responsive to his needs under all circumstances. We will adopt an

approach to this problem which we feel will be different from current systems, and which represents an alternative that will prove beneficial. The difference is in the way we will approach the problem, and in the way in which we control the solution to the problem. We believe our approach will result in a more flexible, extendable, and adaptable system.

Current systems, e.g. the World Wide Military Command and Control System (WWMCCS) and the Naval Tactical Data System (NTDS), approach the problem from the top down. The theory is that if conflicts can be managed at the task force, division, or theater level, then the conflicts will be concluded successfully.

The complexity of such systems is staggering, with tens of computers linked closely together and communicating with far-flung units using radio-frequency bands. We will approach from the opposite direction, with the belief that well-controlled regiments produce well-controlled divisions, and controlled divisions result in controlled armies. Our emphasis is on providing support to the lowest level of tactical commander, whether he is a ship captain or an armored company commander.

Only recently has this need to provide low level support in tactical decision-making been recognized in the military, and many units still have no capability of this kind. Attaining this capability will be a major goal of our design, and this will be evident in the size and speed considerations for the proposed solution.

Additionally, we feel that current systems do not take advantage of the full capabilities of computer systems, and do not use intelligence in the computational manipulation of information. We want to do more than display the information in a way which promotes decision-making: we want to bring the formidable abilities of the machine to bear on the important tasks of analysis and response. One approach to accomplish this is the incorporation of artificial intelligence into the tactical decision-support system (TDSS). Although most systems use intelligence, the explicit use of artificial intelligence techniques has not been executed. We will use artificial intelligence (AI) in our solution.

A very critical, perhaps the most critical, shortcoming of current systems is the difficulty encountered in adapting the TDSS to changing environments. In the practice of war, tactics and weapons technology can change

so quickly that successive battles between forces are quite different. A TDSS, to be truly responsive to the needs of the tactical commander, must be able to adapt to these changes rapidly. Current systems require the development and distribution of new tactics in the form of software packages to adapt to the changing environment. Our proposal is to treat changes to these tactics, and the resultant changes in software flow-of-control, as information which the TDSS manages, just the same as if it were data about the situation. This will, we believe, produce graceful system adaptation in the face of rapidly changing tactics.

Our design is presented at the uppermost logical level, where most of the actual implementation will be transparent to the reader. Thus, we will not delve into the theory of AI nor into that of database management systems, although both are important to the overall design. We will instead concentrate on those areas which are different from current solutions, and therefore less familiar to the reader. At this level, the capability for intelligent and adaptable decision-support will, hopefully, be evident.

Finally, we will make no claims as to the final performance of such a system, although we may at times speak as though such a system exists. It does not exist, and any

claims of either laudable or deplorable performance are matters of speculation.

B. APPROACH TO THE SOLUTION

1. Information, Intelligence, and Decisions

a. Intelligence

For our purposes, we will consider intelligence to be the ability to consider problems and process information to achieve some goal or group of goals, using known resources [Ref 1: p.806]. Thus, intelligence is the accumulation and analysis of information, and the consideration of that information in making decisions. Simply put, intelligence is the use of information in problem solving. [Ref 2].

b. Data

The information which the human gathers using the senses we classify as "data." The purpose of data is to represent the real world, or the environment, to the intelligent agent. For humans, the images produced by the eyes, ears, nose, and the senses of touch and taste are the world with which they deal. The representation of the world in this way is the central contribution to comprehension.

Concepts which can not be represented in this manner are more difficult to understand (e.g. relativity theory and quantum physics).

c. Knowledge

To this data humans apply rules. These rules allow them to use the data in decision-making. Like data, the rules are information, but different in nature. They are obtained not merely by sense, but by learning. Experience, belief, interpolation, and extrapolation are all used by human intelligence to validate rules.

These rules we classify as "knowledge." It is data about data and it is used to understand what data represents and how to manipulate and analyze data to achieve some end (goal).

d. The Intelligent Process

The human intelligent process is the gathering of sensory data, combined with its perception. Perception begins with the formation of a representation, called a percept, from the raw data. The resulting percept may be matched to a representation already in memory, sometimes called a concept. Recognition takes place when a match occurs, and when a match is not made, the percept is not recognized. Knowledge is used in the formation and

manipulation of percepts and concepts [Ref 3: pp. 55-56]. Therefore, the human uses two types of information, data and knowledge, and the intelligent process is impossible without both.

e. Human versus Machine

Although computers and humans are obviously different in nature, the human mind and the central processing unit (CPU) of the computer are similar in many ways. A brief example might suffice. A human job foreman is given the task of allocating his resources to meet some pre-defined objective. He has constraints placed upon him from various sources: all jobs assigned must be accomplished within a certain time-frame; he has a known limit to available personnel; he is expected to get maximum use of his assigned personnel; and job turnaround time is expected to be minimized.

These constraints represent some of the rules under which the foreman operates. The data he uses are the jobs assigned, job requirements, and so on. Using this information, the foreman schedules the work and monitors progress.

We contend that the foreman's task is one which requires intelligence. The analogy to the computer follows

directly from this contention. Since the job scheduler is an important part of any operating system which supports multiprogramming, such operating systems are intelligent.

Although the human and the machine system display intelligence in some common areas, it is apparent that the methods are quite different. Humans display a property which, for lack of a more technical understanding, is called "insight." The ability to focus on the fundamentals of an object promotes efficient storage of representations and rapid access to those representations. Insight, we feel, is a peculiarly human characteristic, which we will not attempt to precisely define.

Some methods for approximating insight, such as the use of dense indexes and heuristics, have proven successful. These methods are appropriate because computers perform simple operations rapidly. If a complex operation is well-understood and can be decomposed into a series of smaller operations, the computer can accomplish the operation very efficiently. Less well-understood problems take much longer and are less efficient.

It is also helpful, in the computer system, to differentiate explicitly between data and knowledge. The line drawn between the levels of information is arbitrary

and arguable, but computers need this explicit distinction to operate efficiently. It should be noted that humans need not make such explicit categorizations and the distinction varies depending upon the problem being considered.

Implementation of our design will make use of "artificial insight" in the use of dense indexes especially, as well as other methods. Although this will result in increased storage (space) requirements, the time saved is considered worth the additional space.

f. The Decision Process

We believe that the process of decision-making adheres to a few fundamental principles. First, information is gathered and added to the store of information already onhand. "Predecisions" are made during this phase, such as what action to take when inconsistent information is encountered. These predecisions are judgements which are made to reduce the ambiguity of the situation represented by the information. Predecision uses knowledge about information limitations, in ensuring that information conforms to those limits. Both data and knowledge may be accumulated during this phase.

Next, the collected information is analyzed to arrive at an accurate summary, or perception, of the

situation. This analysis consists of the consideration of various aspects of the situation, and usually involves a computation to arrive at a summary representation.

Then, the decision process compares the resources used by the various solutions considered during analysis, and compares the results of each of the solutions. Each result will have some relative value to the decision-maker. A good, rational decision would be one in which the resource expenditure and the goal achievement were optimized.

The whole process can thus be decomposed into three phases: acquisition of information; analysis; and decision. The outcome of the process depends on the successful execution of all three phases.

2. The Tactical Environment

For our application area, that of tactical decision-making, our chosen scenario is as follows:

A U.S. naval vessel is at sea during a period of escalating international tension. Information concerning potentially hostile vessels and aircraft is available to the ship's Tactical Action Officer (TAO) from various sources: intelligence sources, his own ship's sensors, etc. The TAO is faced with the task of identifying, by type and threat, a

host of potentially hostile contacts. All available information must be used to the greatest extent possible to maximize the chance of survival in the event that hostilities break out.

Several assumptions are made concerning the nature of the tactical situations which we forsee:

- Peacetime behavior may bear little or no resemblence to behavior during the various phases of an escalating crisis.
- Electronic warfare will be used to deny hostile forces the use of sensors/communications and to ensure friendly forces use of the same.
- Long-range strikes will include combinations of platforms with various levels of intelligent control (cruise missiles, tactical ballistic missiles, manned aircraft).
- Rules of engagement (ROE) will change rapidly during escalation phases of a crisis, probably using combinations of predetermined rules.

We should emphasize that our choice of scenario is purely arbitrary. A land-based missile battery or a squadron of bombers would face similar situations and circumstances. The problem is of a generic nature.

Methods of attack are fast, with little or no warning, and they are deadly, with a single hit powerful encugh to destroy or disable most modern units. Two factors will be most critical in the struggle to survive in a hostile environment: mobility and flexibility. We should have learned from the "great" wars of history that no amount of firepower will offset a weakness in either of these twc areas. The tactical environment is predominantly one of rapidly changing circumstances which affect the various decisions of the commander. On an individual basis, each tactical commander's goal is to win the first battle of the next war. Situational uncertainty will severely hamper achieving this goal.

Sub-sonic, ground-skimming cruise missiles or super-sonic, high-altitude missiles, both equipped with conventional warheads, will probably present the worst threat. In either case, assuming current technology, the time from initial detection of the missile to impact will be approximately seven (7) minutes. Thus, the tactical commander will have a limited amount of time to process a large amount of information and make critical decisions regarding his unit's survival. The commander's most effective use of all of his available resources is largely

dependent upon his overall knowledge of the current situation. The need for accurate and up-to-date intelligence information is magnified when one considers the scenario described above. We must provide the commander this information to assist him in making the correct decisions concerning mobility, flexibility, and survival.

3. Requirement for Adaptability

At this point it must be re-emphasized that the tactical environment is one of rapidly changing circumstances. Military forces in peacetime train to fight the type of warfare which is expected in the next war. Historically, predictions of future conflicts have been poor (at least those by the military leadership entrusted with doctrinal training).

The fruitlessness of some predictions can be gathered from the fact that most of the standing armies of Europe (and the U.S.) had combat cavalry units at the outbreak of World War I, although the principal warfare of that era turned out to be trench warfare. At the beginning of World War II, the concensus among the Allies was that battleships and the Maginot Line would be the predominant factors; of course, aircraft carriers and blitzkrieg quickly ended that thought. We could go on, discussing the

surprises of Vietnam or the Six Day War, but the lesson is already clear: don't become entrenched in the tactics practiced in peacetime.

What is expected in peace may have little resemblence to what happens in war. Secrets are kept which are designed to produce uncertainty, throwing the enemy off-balance long enough to exploit his weaknesses. The inability to adapt to meet unpredictable threats quickly enough is the most fatal flaw a military commander may posess. Conversely, the ability to adapt and take advantage of new circumstances has long been the hallmark of great military genius. It is this adaptability which we feel is lacking in current command and control systems, and which we have included in our system as a salient design feature.

II. GENERAL SYSTEM DESCRIPTION

A. PREFACE

This chapter consists of three separate sections. The first section provides a general description of the major components and characteristics of a typical knowledge-based computer system. This is intended to acquaint the reader with the basic concepts underlying our decision-support system, TAC*. Then, what we feel is a practical and understandable example is given. This example should provide the reader with a basic understanding of the operation of an intelligent, adaptable system. Finally, the basic system configuration and operation of our design is presented.

B. GENERAL

There are many possible variations for a knowledge-based decision-support system, but they all consist of three basic components in one form or another. Such a system may be thought of as being a composition of a database, some sort of control mechanism, and a set of rules [Ref 4: p. 4].

The database (DB) is merely a repository for all of the current data about the environment. It is a collection of

all known facts. As one would surmise, a dynamic environment implies that the DB is continually being updated. Additional information, more recent information, and entirely new information are the three types of data that are stored in the DB. The following example incorporates the three types of information.

Information concerning the retail price of an automobile may be stored in the DB. As information about sales volume becomes available, the inclusion of this data into the DB would constitute additional information. If, due to inflation, the retail price for the automobile is increased by, say $200.00, this new price would represent more recent information. When an entirely new model of automobile is introduced on the market, such information would be classified as new information. The DB does not analyze any type of information, but merely stores it.

The rules may be thought of as being a disjoint set of conditions which have unique responses associated with each. In programming, this is analagous to a series of "IF...THEN" statements. It is through these rules that an understanding of the data stored in the DB is achieved. Like the data in the DB, these rules may also undergo change based upon the environment. Three types of change are possible: addition,

deletion, and revision of rules. New rules may need to be added when a changing environment creates a situation that is not currently covered by the existing set of rules. When the environment changes, some rules may no longer be applicable. Rather than occupying space in the system, these rules should be deleted. Also, different circumstances may trigger a need to revise an existing rule. We will call the repository for system knowledge about the environment a knowledge base (KB). The combination of a dynamic DB and a dynamic KB work in concert to provide a consistent view of the real world for the system user.

The responsibility for insuring that necessary changes are made lies with the control mechanism. All inputs to the system first pass through the controller where a differentiation is made between pure data and rule changes. Data is sent to the DB while rule changes go to the KB. It might be convenient to think of the controller as an interface between the DB and the KB because it is in this component where applicable rules are applied to the corresponding data, changes to both the DB and KB are initiated, and appropriate action (or non-action) is determined. Thus, the controller is the heart of the knowledge-based decision-support system.

The preceding section has given a very basic overview of a knowledge-based decision-support system, describing the three major components. Although such a system is much more complex than this summary indicates, especially when one considers the possible system variations, an attempt was made to lay the foundation for understanding the detailed explanation of our prototype system, TAC*, which follows in Chapter III.

C. PRACTICAL EXAMPLE

As an example of the operation of an intelligent and adaptable system, we will consider a professional football team. This example is understandable for most people and also has a direct analogy to the operation of our decision-support system. In the tactical environment, the overall goal is to win the battle. Similarly, on the football field, the goal is to win the game.

1. Resources

Because the total number of players a team may have is strictly controlled, there is a definite limitation to the coach's available resources. However, the athletic potential, or capabilities, of all players is not controlled nor are these capabilities identical. Therefore, the overall quality of a team's resources will vary from one

professional organization to another. The coach must attempt to win each game based upon these number and quality constraints.

2. Game Preparation

Football teams practice in preparation for actual games. The overall game strategy and specific plays are developed based upon what is expected to occur during the game. Thus, a certain amount of prediction is necessary in preparation for the game. The coaches are trying to predict (into the future) those plays that the opposition will try.

Before each game, the opposing team's strengths and weaknesses are studied. Data about the other team's tendencies is accumulated and analyzed, thus giving the coaching staff some knowledge about the opponent. Eventually, the coaching staff arrives at conclusions as to what the other team is likely to do in specific situations throughout the game. (Since each team has different strengths and weaknesses, these conclusions will change from game to game.) The conclusions are then written down in the form of plays that the team will use during the game. This sequence of plays is called a "game plan." In essence, the plays are the rules that the team will follow throughout the contest. There is another type of rule which the team

follows in both the preparation for and the actual play of the game. These are the rules of the league. For example, each team may only have eleven players on the field during the execution of a play, an offensive player may not block a defensive player below the waist, etc. Both types of rules are combined and determine the flow of action during the game. Incidentally, those league rules are enforced by the referees in an actual game. Generally speaking, the referees oversee all action on the field. When a team's action or formation conflicts with the league rules, the referees assess a penalty against the offending team and, thus, resolve the conflict.

In summary, then, at the opening kickoff, each team has accumulated data about the other team and each has been able to gain some knowledge about the other based upon the analysis of the data.

3. Playing The Game

At the outset of the game, a team wants to follow the overall strategy as defined in its game plan. For example, in a particular situation, the plan may call for the quarterback to throw a short pass to the tight end ten yards from the line of scrimmage and near the sideline. However, the coaching staff realizes that their opponents

may try to win the game by making changes to their own plays during the course of the game. These changes may render a particular play ineffective. For example, the opposing team may decide to have two defenders play against the tight end, instead of one, and the sideline pass mentioned above may not be able to be completed with the new defensive coverage. Therefore, the coaching staff will have several personnel positioned high above the field to detect any such changes in the defensive coverage and suggest alternate plays to call which will be effective against the new defense. A good choice might be a play run earlier which was successful against the opposition's present defense. This would entail projecting back in time to find the successful play. This group of people monitors the action on the field, but becomes the controller of the action when changes occur. In essence, the group determines when the game plan is no longer effective and tries to adapt to new situations by changing the rules that their team will follow during the remainder of the game. Further, their recommended changes must be made as quickly as possible to preclude the opposing team from dominating the action and ultimately winning the game.

4. Summary

A successful professional football team may be characterized as being an intelligent organization which utilizes data to derive knowledge about its opponents and an organization which is capable of changing the rules under which it operates when the situation changes. The catalyst which insures that the team is rapidly adaptable is the group of controllers positioned above the action on the field. While this group selects a particular play that will hopefully be successful, the quarterback makes the final decision as to which play is actually to be run. For example, if the group tells the quarterback to run play "x", and when the team deploys at the line of scrimmage, an unexpected defensive alignment may be applied by the opposing team. So, the quarterback will change the play to be run by calling a special set of numbers at the line of scrimmage which signify play "y."

The next section should reflect a remarkable similarity between the operation of a professional football team and the operation of our tactical decision-support system.

D. TAC*: BASIC DESCRIPTION

Essentially, TAC* is an intelligent computer system which utilizes current information about the real world, stcred in its dynamic "infobase", to understand changes which occur in the real world. Besides the routine changes which occur due to the passing cf time or the movement of units, a change may also signal a "conflict" between data items and knowledge rules. A conflict is an inconsistency requiring a decision by some agent which results in a consistent database. A conflict may be classified into one of three areas:

1. Those that occur due to technological or infcrmational limitations; e.g. Two known ships pass within "x" meters of each other. Both ships are being tracked, however there exists an area wherein it becomes impossible to differentiate between these ships. This resolution limitation of our current technology poses obvious problems in determining which ship is sailing what course as both emerge from the ambiguous area.

2. Those that require Fcrward Prediction; e.g. Intelligence sources are tracking a flight of known hostile planes when, suddenly, the flight disappears. It is imperative that we be able to project forward in time giving

the ultimate target, or at least the heading, for this flight of hostile planes.

3. Those that require Backward Prediction; e.g. Intelligence sources report a troop build-up of known size has occured during the past 24 hours in Eastern Europe. We must ascertain whether the force is mechanized or armored, which other units might be in support, etc. Backward Prediction allows us to project backward in time to a verified, real-world situation in order to ultimately identify the unit in question. Time, distance, overall logistics capabilities, available means of movement, etc. must be considered.

As can be seen from the above examples, TAC* should prove to be an invaluable tool in both tactical and strategic planning, with specific applications for all branches of the military services.

The three major components of TAC* are the controller (CCN), the database management subsystem (DBMS), and the knowledge base management subsystem (KBMS). The controller may be thought of as a "world watcher" with the responsibility of taking control of the system when a conflict arises or information changes are needed. Direct interaction with the database manager (DBM) insures that the

results of all updates to the database are known by the controller, thus insuring that all conflicts are detected. Corresponding to the database is a knowledge base which, in essence, is a set of rules that the database must abide by; i.e. an armored brigade cannot move 1000 miles in a six hour period, or an aircraft carrier is not capable of making a 180 degree turn within a 200 meter space. Such restrictions might be termed "data constraints."

This brief overview of TAC* should be adequate to proceed on a common level and, hopefully, will enable us to understand the detailed system description which follows. Then we will be able to focus our attention on specific solutions to specific problems.

III. DETAILED SCHEME

A. REAL-WORLD REPRESENTATION

The purpose of TAC* is to analyze the real-world and, based on rules provided to the system, to advise the decision-maker on a course of action. To do this, TAC* must be able to understand the real-world. The real-world is represented to TAC* by the database (DB). A DB is a series of records which contain information about objects and their relationships with other objects.

A record is a structured collection of data, each substructure of which is called a "field." These fields represent abstract properties of an object, called "attributes." Each attribute has an associated value. The fields of a record contain values for the attributes of the object which is represented by that record.

1. Areas of Interest

The example we have chosen is of a naval vessel on the open sea (i.e. no land mass). The area surrounding that vessel is its "area of interest" (AOI). An AOI represents a section of the real world. Position within an AOI will be represented using a coordinate system. Notice that the

definition of AOI depends upon the point of view (tactical responsibility) of the user command. For the Joint Chiefs of Staff, the entire globe may be their AOI. However, for a single small unit, the AOI will generally correspond to the area in which the unit is operating. The size of the area will depend on the capability of the unit and the limits of the defense perimeter.

2. States

The condition of a system can be described in terms of its "state." State is the term which we shall use to identify the condition of the TAC* system, the real-world situation represented in the TAC* DB, and the individual object-records in that DB.

An object state will be that combination of record fields which describes the status of the object. For most objects, this will comprise the position, identification, and warfare status fields. Certain rules in the KB will pertain to allowable object transitions from one state to another. Only transitions which obey these rules will be allowed to occur automatically. Transitions violating these object transition rules may be held pending and made known to the system operator. Only valid object state transitions will be permitted.

An area state value describes the evaluation of an area of interest. Area states take on values which represent a tactical view of the AOI. Ship types and capabilities, relative positioning and political situations all contribute to the evaluation of an area state. Area states are summaries of all of the local (object) states within that area. A change to a local state may affect the area state, but an area state cannot change without a change to an object state.

Area state values are determined by object states. They are computational and analytical summations of the set of objects included in that area. The area state value is determined following each valid DB update. It is this area state which the system uses to determine what action, if any, to take.

The system state takes on values which describe the current system status. For example, the system may be "idle" with no pending operations. Or it may be updating the DB, or analyzing the results of an update, or updating the KB.

We avoid the use of local and global states to avoid ambiguity. Instead, we will consider object states and area states and their associated transitions. Depending upon the

sccpe of the system and the rules in the KB, an object in one system may be an area (AOI) in another (smaller) system. In our example, we deal with a single ship's system, and therefore the objects are other individual platforms and the area is the ship's AOI.

B. DETAILED MODULE DESCRIPTIONS

1. Interface

a. Input

Our system is essentially symbiotic in nature, relying upon raw information from other sources to feed its database. In order to do this reliably, an input subsystem with appropriate capabilities must be used.

The system is message oriented and designed to operate with multiple inputs and with multiple priority messages. Three major source classes for messages must be considered: system and operator generated, intrinsic sensor generated, and externally generated.

System and operator generated messages are the easiest to handle. System messages occur only as a result of some state change. When a new state value has as its action a data or rule transaction, the system generates a message which is forwarded to the input spooler (via the output spooler). Next, the operator may enter a message to

the system. Most KB changes, and some DB update/request messages will be created in this way.

The next most important general source class is intrinsic sensor. Sensing devices on the user platform will have a message formatting translator to allow direct input to the input spool. These inputs from ship sensors (radar, sonar, ESM) will be multiplexed and queued according to priority.

The final major source class is externally generated reports. The best examples of these are intelligence reports, OPREP-3 messages, NTDS Link 11/14 type intercommunication, and UNITREPs. All of this information originates outside the ship and is received via radio receiving units. These radio messages must be decrypted/translated and channelled to the input spooler. A separate processing system may be needed to do this in order to insure rapid access to external data.

All of these sources meet at one destination: the Input Spooling Process (ISP). The purposes of the ISP are:

- To collect, merge, and sort the incoming data quickly.

• To act as a buffer during high volume traffic periods.

• To feed transactions one at a time intc the Input Selector Switch.

The ISP may be considered to be a smart queuing device. If it is expected that large amounts of data will be incoming, the system may schedule the ISP more frequently; more sensibly, the system design is amenable to some multiprocessing and the ISP could easily use a dedicated processing unit.

b. Output

Like the Input Spooling Process, the Output Spooling Process manages the flow of information out of the system. Output destinations fall into four major categories: operator, system, intrinsic, and external.

Operator messages are advice or orders to the operator. The result of some rule invocations may be to inform the decision-maker of a new level of readiness which must be achieved. Or it may notify the operator that an invalid object state transition was attempted, and ask for instructions to complete the pending action.

System message outputs, as mentioned above, allow the system to feed back to itself. Some area states

might require this, or a change in a KB item affecting the current state may produce this result. It should be considered an infrequent action.

Correlation of two objects in the database may result in intrinsic sensor action. The need for additional or more detailed surveillance information could also cause the system to send a message to a sensor station. For example, a report received on an unidentified subsurface contact may neglect to include a depth. TAC* may then prompt the sonar team for this information, or request a change in mode to determine depth, or estimate it for them based on current sea conditions.

In certain situations, it may be advantageous for TAC* to generate messages and send them directly to the Navy Telecommunications System for broadcast. Periodic situation summaries or high-priority operational reports could be easily handled by TAC*, enabling the tactical commander to concentrate on the situation at hand. Naturally, this capability could be modified to allow human intervention and editing prior to transmission, and this ability could be disabled during those periods of emission control when transmission is not desired.

As with input the system will handle output communications with spooling. An Output Spooling Process (OSP) would receive and forward output messages to their destinations. Multiple spooling by class would be used to ensure quick response. One spooler would handle the operator interface, another the intrinsic sensors, and another the external broadcast channel. A separate spooler is not needed for the system messages, as the ISP will spool that class of message anyway. The OSP Intrinsic Sensor Spool could also execute the demultiplexing process.

2. Control Subsystem (CON)

The Control Subsystem acts as the highest level of control for the TAC* system. Input to and output from the TAC* system are through the modules of the Control Subsystem (CON). CON has two basic functions:

- To decide the destination of an incoming message.
- To decide the destination of outgoing actions.

CON has four components: the State Comparison Module (SCM), the Log Recorder (LOG), the Response Driver (DRIVER), and the Input Selection Module (ISM). The functions of each are described below.

a. Input Selection Module (ISM)

The Input Selection Mcdule (ISM) is the gatekeeper for the TAC* system. Messages are received from the Input Spooling Process and the determination is made as to whether the message is a data transaction (DTX) or a rule transaction (RTX). DTX traffic is routed to the DBMS Subsystem via the DBMS Ingate. RTX traffic is routed to the State Comparisen Module.

Selection in the ISM is accomplished by a simple boclean (bit) check. A message is either a DTX or an RTX, but nct both. A DTX message is placed in the DBMS Ingate, which acts as a mailbox. RTX messages require different handling, and are first checked by the State Comparison Mcdule.

b. State Comparison Module (SCM)

The State Comparison Module (SCM) alsc performs selection, but this is more complicated than the ISM. First, the SCM compares the input state to the current area state of the system. Based upon this comparison, and the knowledge of where the input state originated, the SCM decides where to send the message.

If the input is from the ISM, and the states do not match, the message is a rule change requiring no special

handling, and it is passed to the KBMS for action. If the ISM is the source and states are the same, the message is an RTX affecting the current state, and is sent to the Response Fetch module of the KBMS for immediate handling.

The input may be from the DBMS. Specifically, the Area-State Monitor will send its latest area state summary to the SCM. If states do not change, the message is simply discarded. If states do change, however, the message is passed to the Response-Fetch module for matching and reaction, and the current state is updated.

Thus, the State Comparison Module performs quick matching to enable rapid system response to new rules or new data.

c. Log Recorder (LOG)

Another function of the CON is to record the effects and transactions on the system. This involves both object state transitions and area state transitions. Recording this information allows for smooth recovery of the system from crash, from a cold start to a consistent, accurate DB.

If properly used, the LOG can also check the effectiveness of rule actions. Programs could be written which would scan the Log Record and compile statistical

performance figures on each rule, thereby allowing the pinpointing of faulty assumptions and the correction of poor performers. During development and tuning of new KB's, this could be very important, possibly allowing the test of new rules prior to distribution to operating forces.

d. Response Driver (DRIVER)

The Response Driver (DRIVER) is the heart of the control sub-system. While the KBMS acts as the repository for knowledge, the DRIVER implements that knowledge. It receives messages from the KBMS (Response Fetch) and from the DBMS (Object Transition Validation or OTV) modules and converts them to inquiries, advice, or orders.

From the KBMS Response Fetch module, it gets the result of rule-condition matching, which produces the reaction part of the rules stored in the system KB. These reactions it converts to device orders or advice to other systems. It may also tell the OSP to send a message to the ISP as a data transaction. These possible actions are the programmed responses to an area state summary.

The DRIVER may also get object transition messages from the OTV module of the DBMS, signalling that a data transaction is being held pending due to a conflict between the DB and the object KB. This prompt is passed to

the user to allow him to choose the method of resolution, or the DRIVER may be programmed to make the decision. The chcices and the methodology of the OTV will be discussed later.

The DRIVER is the connector to the output for the entire system. It is this module which permits TAC* to influence other systems and provide tactical advice.

The function of CON is to pass information to and frcm the KBMS/DBMS subsystems, and tc provide the framework arcund which TAC* executes. As with the Input and Output subsystems, it is possible that large TAC* systems would benefit from a dedicated CON processor which would be tightly coupled to the INPUT, OUTPUT, DBMS, and KBMS prccesses. For small systems, this should not be needed. Next we look at the Database Management Subsystem (DBMS).

3. Database Management Subsystem (DBMS)

The next major segment of TAC* is the Database Management Subsystem (DBMS). It consists of six majcr modules which operate together to represent a real-world situation in data records. The parts of the DBMS allow the representatiocn to be created, to grow, to be changed, and to be accessed.

a. Ingate

The Ingate acts as the interface between the DBMS and the input of the system. Recall that the Input Selection Module (ISM) of CON sends incoming data transactions to the DBMS. These transactions are sent to the KBMS via the KBMS Ingate.

The Ingate is a buffering and queuing process which allows the DBMS and ISM to operate at different speeds. Incoming transactions are queued and processed in order. A pre-emptive queue may be used to ensure that the most important updates get top priority. Thus, a high precedence message would be expected to go to the top of the queue, while low precedence messages would go to the bottom. Alternatively, an ordering of four seperate queues could be used. When the DB Management Module (DBMM) is ready for the next message, it gets the message from the Ingate.

b. Database Management Module (DBMM)

The Database Management Module (DBMM) acts to interpret DB transactions and execute the required change orders. Database Management System (DBMS) theory is well known, and we will not describe the DBMS operation in detail.

To update a record, the DBMM gets that record from the DB. A lookup must be done in the DB Index, a data structure which is maintained by the DBMM. To promote fast access, we envision a well-structured, dense index. This will require more storage than that required otherwise.

Once the record is brought into main storage, a quick check must be made to see if the pending transaction is the most recent transaction. If it is not, the pending transaction is entered into the archival portion of the record and the record is put back into the DB. If a pending transaction is more recent than the most recent already in the current record, the pending transaction replaces the older transaction, and the older transaction is placed into the archival section. Prior to a "put", pending transaction results are checked by the Object Transition Validation Module (see below).

Additions of new objects and deletion of old records is also done by the DBMM. These operations are relatively simple as they only invoke the Index and the Free Record Queue, both of which reside in the DBMM. It is interesting to note that to be efficient in storage, the DB Index and Free Record Queue may themselves be records in the

DB, and only be brought into main storage when needed for lookup and management.

Many ways of indexing and structuring databases exist, and we will make no attempt to explain all of them nor will we limit the applicability of our system to one specific type. We feel that the structure of our system permits independent (or nearly so) operation of the DBM and KBM sub-systems. These managers are isolated from the mechanisms by the message-handling modules of the system.

c. The Database (DB)

The Database (DB) is the collection of records which represents the real-world to the computer. The DB records contain information about objects in the world and the relationships between these objects. Data about each object is divided into two major categories: current and archival. The current section is itself a record. It defines the current state of a physical or abstract object in the database. Examples of current data are: the last known position report, with asscciated effective time-stamp; the generic type identification; and the warfare status. Warfare status will define the intrinsic capabilities of that unit (weapons on board, damage sustained, etc.).

The archival section is a record of other pertinent information about an object. Past position reports, non-tactical information, and other non-priority data will be retained in the archival section.

d. Object Transition Validation Module

When an update is received for a specific DB object, certain validity checks may be performed to ensure that data received is consistent. The Object Transition Validation (OTV) Module does this by comparing the proposed change to physical or behavioral rules which are recorded in the knowledge base. A data transaction which conflicts with a rule of the system will be identified and corrective action may be taken.

For example, a data transaction reports that object "x" is now 100 nautical miles (nm) from where it was one hour earlier. Object "x" has been evaluated previously as a destroyer-type ship. Now the data transaction conflicts with the maximum possible speed for surface ships of this type, which is 35 knots. Either the data transaction is incorrect, the old report was incorrect, the evaluation was in error, or else the rule limiting surface speeds is wrong. Such conflicts must be considered when designing a system with imperfect sensory data and imperfect knowledge.

The OTV Module can handle such a conflict in several ways. It may allow the data transaction to be made and note the existence of the conflict. This would be called "data supremacy" because the information included in the data transaction is presumed to be correct. Conversely, the OTV Module may reject the data transaction, and merely note that the transaction was attempted, but not allowed. This would be called "knowledge supremacy" because the information included in the knowledge base is presumed to be correct.

Still another alternative would be to hold the transaction pending and inform the operator of the conflict. The human could then make the decision as to which information item was incorrect, and direct the completion or abortion of the transaction. Another possibility is a hybrid version which considers the relative amount of error, and permits "small" conflicts to be transacted. The possibilities seem limitless, but all will depend upon the supremacy of data, the supremacy of knowledge, or the equality of data and knowledge. Under any circumstances, each transaction will either be completed or aborted.

e. Area State Monitor (STATE_MON)

So far, we have just considered the objects in the database. Now we must look at the DB as a whole, representing the state of the real world as modelled by the computer. What we need from the DB is a statement which summarizes the conditions which exist in reality. The Area State Monitor performs this task.

The Area State Monitor (STATE_MON) considers the presence of physical objects in the database, and also the values of certain abstract DB objects. For example, the abstract DB object "Readiness_Condition" will have a value dependent upon the declared state of readiness as promulgated by the National Command Authority (NCA) or local area commanders. Numbers and types of hostile units and friendly forces will also be considered.

From this data, the STATE_MON extracts a value which represents the state of the area of interest (AOI). This area state summary is what TAC* uses to determine what reaction is needed/recommended. The STATE_MON computes this summary and passes it to the Outgate for forwarding to the Log Recorder and State Comparison Module. The STATE_MON uses data and knowledge to derive the summary value describing the area state. It has direct access to the KBMS

Area Analysis Rule Subbase. These rules may be changed like any other item in the KB, resulting in adaptable data analysis.

The STATE_MON is perhaps the most important of the TAC* modules, because it performs the actual analysis of situational data. A sub-base of the KB will be accessed by the STATE_MON, and rules directing the computational analysis of data will be called and executed by the STATE_MON.

In the football analogy, the function of the STATE_MON is performed by the group of controllers from high above the field and by the quarterback as he steps to the line of scrimmage. Scanning the field of play, he notes the disposition of the offense and defense, looks for "key" indications of defensive intentions, and draws a rapid conclusion. For the QB, that conclusion is whether to execute the play called in the huddle, or to call a new play at the line of scrimmage. Final play selection depends almost entirely on the analysis by the QB, and he typically has about 10 seconds in which to make his decision.

In TAC*, the STATE_MON acts like the QB in analyzing force disposition, capabilities, and signs which might indicate enemy intentions. It does this

computationally, by deriving a value for the Area State Summary. The Summary is a group of bits whose value depends upon various aspects of the state.

Without actually defining a function for the STATE_MON, it may be hard to understand, but the analogy to the quarterback is very accurate. Detailed operation is dependent on the implementation, and the implementation of the Area State Monitor will be one of the most difficult tasks in any TAC* implementation.

f. Outgate

The Outgate is the module within the DBMS which passes messages to the other parts of the system. The Outgate receives traffic from the DBMM and the STATE_MON. From the DBMM, the Outgate gets the transaction order, and from the STATE_MON, the Outgate gets the Area State Summary.

The data transaction and the resulting Area State Summary are paired together at the Outgate. The resulting message is then sent to the Log Recorder for inclusion in the DB LOG Record. The Area State Summary is also sent from the Outgate to the State Comparison Module.

The DBMS as described above is straight-forward and could be generalized to any database system. It uses knowledge to check for the validity of object transitions,

thus ensuring a valid DB at the object level. Note that the validation of object transitions is considered sufficient for the validation of area state transitions (this is stated without formal proof). Other than the above-noted special functions of the DBMS, it works essentially as a classical DBMS.

4. Knowledge Base Management Subsystem (KBMS)

The Knowledge Base Management Subsystem is the final major segment of the TAC* System. The KBMS is the repository and manager for the knowledge that TAC* posesses about the real world. Most of its functions are analogous to those of a classic DBMS, but there are some differences.

a. Knowledge Base Management Module (KBMM)

The Knowledge Base Management Module (KBMM) is that part of the KBMS which performs the DBMS-like functions of getting, updating, and putting records in the Knowledge Base (KB). Messages containing updates/changes are received by the KBMM from the State Comparison Module in the Control Subsystem.

The KBMM takes these change messages (Rule Transactions) and implements them on the KB. This involves first getting the record containing the rule from the KB file. Changes are then made to the rule according to the

instructions of the RTX message. A rule may be deleted, added, or its associated response altered. Once this is accomplished, the record is placed (PUT) back into the KB and a log message is sent to the KB Log Recorder for temporary storage until the next KB dump.

b. Response Fetch Module (RFM)

The Response Fetch Module (RFM) of the KBMS is used to get the appropriate Rule Response from the KB and send it to the Response Driver in the CON Subsystem. The input to the RFM is either a rule transaction or an Area State Summary message from the State Comparison Module. If the input is a rule transaction, then the associated Response is attached to the Rule. The Response is copied, checked, and sent to the Response Driver.

The other type of input is an Area State Summary, and when a summary is received, the RFM uses the STATE value to directly access the required record. After the record is read, the Response is sent to the Response Driver. Accessing the Response should be fast and easy, using a dense index in the KB.

c. The Knowledge Base (KB)

The key to the capability and adaptability of the TAC* System is the notion of a Knowledge Base, where the

relations and implications of what the programmers have "taught" TAC* about the real world are stored. The Rules of the KB are things which TAC* knows or expects to be true, along with some consequent results.

To promote fast response and efficient use of storage space, the KB is divided into at least four major parts: the dense Index; the Object Rules; the Area Response Rules; and the Area Analysis Rules.

The Index organizes the physical data into a useful structure, acting as a dense index into the relatively sparse Area Response Rule section, thereby permitting better use of storage. This is necessary to prevent the need to search the KB to locate the required record, or else to have a huge file section, much of which would not be used (being empty or replicated data).

For example, if we used a 16-bit Area State Summary, the corresponding number of different states would be 65536 (64K), 2 to the 16th, too large for a small computer, even with a hard disk mass storage, since a record would be needed for each state. Within the access functions of the KBMM and the RFM, the Index is used to map the 16-bit number to a much smaller number of records, with the mapping being faster than an explicit search. The relation between

Area State Summary values and Area Response Rules will be many-to-one, with relatively few Response Rules.

An Index for 64K Area States would require about one Megabyte of storage for the Rule (key) list alone, with somewhat less than that for the Response Record list. Since hard disks are now availible with 5 to 20 Megabyte storage capacity, this seems well within the range of microcomputers. This would leave the bulk of mass storage for the larger Response records.

The Area Rule Response section of the KB contains the reactions of the system to a specific value of the Area State. It is easy to store these reactions in contiguous records of fixed length, with some breakage expected. Actions may be messages to the operator, advice/orders to remote stations, or data transactions which feed back into the system. All actions are executed by the system via the Response Driver and the Output Spooling Process as described above.

The Area Analysis Rules are also included in the KB. These rules may be considered to be analogous to procedures, in that they enable the State Monitor to compute a State Summary value. The State Summary value is an integer which is the result of the operation of the

STATE_MON, and it sufficiently describes the state of the AOI. It is used by the KBMM to index into the KB Response Sub-base. The operation and use of these rules is described in the section on the Area State Monitor, above.

The final section of the KB is allocated to Object Rules. Object Rules are used by the Object Transition Validation (OTV) Module of the DBMS to validate state changes resulting from data transactions. The Object Rules represent physical constraints which limit the ability to transform from one state to another (maximum/minimum speeds, depths, altitudes), as well as known political constraints (e.g. units of type "X" are not permitted to operate in this area). Depending upon the amount of expertise the system is designed to have, the Object Rule KB may be quite large and require its own index and access mechanisms. The KBMM can, however, manage changes to both the Object Rules and the Area Rules quite easily.

This completes the detailed decription of the TAC* System at the functional level of the major modules and subsystems. To futher understand the system it might be helpful to trace two transactions through the TAC* process.

C. EXAMPLES OF SYSTEM OPERATION

1. Data Transaction (DTX)

The Input Spooling Process (ISP) receives a message from the ship's Communication Center as the result of an Intelligence Report fro the Fleet Intelligence Center. A data transaction (DTX) is sent to the ISP which states that a Soviet Navy AGI (intelligence gathering ship) is operating in your area. The DTX is queued in the ISP until the CON Input Selector is not busy.

The Input Selector Switch (ISM) receives the transaction, sees that it is data, and sends it to the DBMS Ingate, where it is again queued. When the DBMS signals to the Ingate that it is ready for the next message, the DTX is forwarded. The DBMS determines that the DTX is an object transaction, and it calls the Object Transition Validator to ensure correctness of the change.

This particular transaction reflects a change in position of the AGI of about 50nm in 2 hours, for an average speed of 25 knots. The OTV module, accessing the Object Rules KB, knows that the maximum speed of an AGI is only 20 knots. The OTV sends a message to the operator noting the discrepancy between rules and data, but it is programmed for data supremacy; thus the DTX is allowed. The transaction is

sent to the Outgate, where it is coupled with the Area State Summary.

The Area State Monitor (STATE_MON) determines that a change in the AGI's positicn changes the Area State. It calculates the new Area State Summary, and sends it to the DBMS Outgate. At the Outgate the Area State Summary is sent to the State Comparision Module and the combined DTX/Area State Summary is sent to the Log Recorder.

The SCM compares the incoming state with the old one and determines that a new state exists. The new state is sent to the Response Fetch Module, where the correct response is accessed and forwarded to the Response Driver. The Response Driver determines that the destinatica of the response is the operator, and it sends the message to the Output Spooling Process (OSP). The OSP sends the message to the TAO, who reads it and notes that the recommended response is to set a special Emission Control posture to restrict the amount of technical intelligence availible to the AGI. TAC* also advises the TAO to alert his own TECHELINT collection team, and specific emitters of interest on the AGI are noted.

The reaction of TAC* to the DTX is now completed, and the TAO has received scund advice. Note that TAC* also

warned the operator of possible inaccuracy in the data when the OTV did not completely validate the transaction.

2. Rule Transaction (RTX)

Shortly after the processing of the DTX above, another message is received in the Communications Center. The Fleet Commander has ordered that all AGI's of a certain class be closely investigated for new capabilities. The AGI in the AOI is one of this type.

Up to the Input Selector, the flow is the same, but now the message is sent to the State Comparison Module because it is a Rule Transaction (RTX). The ISM also sends the RTX to the KBMS which quickly enters it into the KB. At the SCM, it is determined that the new Rule affects the current area state. The transaction is sent to the Response Fetch Module for action.

The RFM accesses the Response part of the transaction, and formats it for output. It is then sent to the Response Driver, which in turn sends it to the OSP for reporting to the operator (or the TAO). The TAO now is aware that a message has been received directing intelligence gathering action against the AGI. Appropriate steps are listed to put the ship on its maximum readiness footing for the coming task.

TAC* again has reacted exactly as desired in response to a change in knowledge. The ability to quickly react to the change is crucial, as failure to do so may have fatal results in the tactical environment.

IV. DESIGN ANALYSIS AND RECOMMENDATIONS

A. PREFACE

When analyzing any design, it is important to consider the original goal and determine how closely one has come to achieving it. Our specific goal was to design, at the conceptual level, a computerized decision-support system to assist tactical commanders in the decision-making process. We feel that TAC* is such a design. However, with any new system, both advantages and disadvantages are accrued, and the TAC* system is no exception. Therefore, this chapter is an attempt to impartially consider both positive and negative aspects of our design. Such areas as applicable domain, modular design, real-time response, and system intelligence are examined. We will also discuss those system extensions which, in our opinion, are feasible and would be relatively easy to implement. Due to continuing progress in areas such as magnetic bubble memory, 16-bit and 32-bit microcomputers, and systems of computers, we contend that our "projections" of TAC*'s reliability should be regarded as more than mere speculation. We do accept as fact, though, that the ultimate test of reliability lies in the successful implementation of our design.

B. ADVANTAGES OF TAC*

A TAC* prototype system can offer the lower-level unit commander an additional tool for use in his decision-making process. This system is applicable to the tactical environment and has the capability to intelligently process large amounts of data in real-time.

1. Applicable Domain

TAC* is a knowledge-based decision-support system. We believe that such a system is applicable to a specific domain when the following criteria are met:

- There exists a large amount of information about the specific domain.
- Such information is capable of being decomposed into either data about the domain or knowledge about the domain.
- The control system strategy is adaptable to computer operation in "domain real-time."

There is general agreement that the tactical environment generates large amounts of information from higher-level commanders, from active and passive intelligence gathering sources, and from changing environmental conditions (to name just a few). Information from any of these origins may easily be represented as

either pure data, which is stored in a database, or rules to which the data must conform. Due to the relatively small number of available options for a particular tactical situation (current target, available weapons systems, etc), the system control structure will be able to operate efficiently. Thus, a knowledge-based decision-support system is, theoretically, well suited for application in the tactical environment. Implementation and utilization will be the basis for determining if such a system is realistically applicable in a tactical situation.

2. System Intelligence

A primary goal behind the development of TAC* was to devise a system capable of assisting the tactical unit commander in his decision-making process. As such, the system first had to be intelligent. That is, it had to be capable of storing large amounts of data, knowing what that data meant, and how it could be used. System intelligence had to be intrinsic, rather than provided by the human element. By correlating applicable rules and data, intelligence has been designed into the system. Processing data, analyzing changes, and formulating alternative solutions can be extremely time-consuming for the human, but should be accomplished by TAC* in less time. The ultimate

benefit gained will be to allow the commander more time to devote to those tactical considerations which cannot be left to the computer.

We believe that machines should not be considered as replacements for humans. Rather, the effectiveness of combining human and machine intelligence should be maximized. TAC* should allow the computer to perform those functions which are time-consuming and prone to human error, while preserving the human factor in areas of judgement and experience. Such integrated intelligence maximizes the effectiveness and efficiency of the decision-making process.

We feel that the TAC* system is a logical application of decision-support principles to the area of tactics. While NTDS gathers information and presents it to the human operator in a way that promotes analysis, TAC* will do much more. TAC* will analyze the situation that the data represents using information stored in its knowledge base and recommend responses to the operator. TAC* will not be a glorified information-handling system; it will be an expert in the area of tactics and weapons employment in the tactical environment.

The obvious question is "Why not extend the present system rather than introducing a new one?" We contend that NTDS cannot be extended to achieve the speed of learning which our system will offer. This is because NTDS is a static software package; once a version is loaded, system response is fixed until an updated version is distributed. On the other hand, TAC* should learn a completely new tactic in a short period of time by simply updating its knowledge base.

3. Distributed System

A fully distributed computer system is one in which the hardware (HW), software (SW), and data are resident at various local sites. It is, therefore, a "stand alone" system which, if part of a network, communicates with other sites via messages. The characteristics of TAC* classify the system in this distributed category. One of the most important benefits derived from this type of system is the availability of a complete computer system which can be devoted entirely to the needs of the local site. In our tactical environment scenario, this means that the local commander does not have to compete with other commanders (outside of his Area of Interest) for the use of the computer system.

Given the alternative between a distributed or a centralized system, the former was chosen for TAC* even though traditional military control of subordinate units has been extremely centralized. As specified in Chapter I, we believe that the independent operation of tactical units in any future conflict will promote better control up, rather than down, the chain of command. This is not to say, though, that higher level strategic decisions will not be forthcoming in the form of specific orders. Whether tactical independence is recognized and planned for by the command structure or occurs, by necessity, after the fact is immaterial. We believe that independent unit operation will be unavoidable and the distributed system will be the principal factor responsible for survival in the tactical environment.

Planning a system which supports independent unit operation provides the most flexibility to the tactical commander. However, the tailoring of a "stand-alone" system does not preclude its use in centralized command structures. Rather than a limited applicability, TAC* should be equally applicable in either centrally controlled or independent, isolated scenarios. Consider the following situation:

A tightly controlled network of computers is devised to command a tactical group in battle. In the first minutes of hostile action, the Net Control Station is put out of action by enemy fire and electronic jamming on the digital data link is severe. Local tactical units, their sophisticated centralized computer systems rendered useless by the confusion, damage, and jamming, overwhelm the commanders with false and superfluous information. In the next few minutes, the enemy launches a major offensive strike and severe losses to friendly forces are suffered.

Because such an occurence is possible, we designed a system capable of either independent or group operations. An Input/Output system based on message-passing was included for efficiency and flexibility. TAC* was conceived as a modular system, with each module or node performing a specific function. A group of nodes, communicating by means of messages, would constitute one complete system called a "cluster." Nodes will perform the functions of Input, Output, Control, DB Management, and KB Management. Clusters will be capable of inter-communication by means of messages passed between respective I/O nodes, using any bus protocol deemed appropriate, thus forming a network of clusters. We believe this is this is the best way to achieve reliability and modularity at the least cost.

4. System of Computers

The design of TAC* has been highly modularized to allow for the flexibility of implementing it as a system of microcomputers, rather than using a large mainframe. Instead of devoting a fixed amount of core memory for the DBMS, another fixed amount for security and protection, etc. one small computer may be used for each major function or module. For example, the Area State Monitor could be configured to have its own set of rules which it would use to derive the Area State Summary. The rules would reside in one microcomputer whose sole responsibility would be to the Area State Monitor. In essence, such a configuration could be viewed as being a network within a network. The inner network would be exclusively devoted to the tactical unit, with the various unit's systems combining to form a "global" network. Should one of the machines in the inner network malfunction, only a portion of the system's total capability would be lost. Further, system maintainability and the isolation of component failures should be more simplified than those associated with a large, centralized system. Additionally, the use of a simple control structure with a small amount of intrinsic knowledge should improve the chances of a hardware deficiency being repairable by "plug

and go" strategies. Since nearly all of the system control information is included in the KB, should any of the several CPU's fail, replacing it should be straightforward and easy.

Note also that the mechanism of the control structure is simple and consistent. Adaptability is achieved by changing KB entries dynamically rather than by complex control structures. These simple control mechanism instructions could be programmed in ROM-type storage to prevent accidental erasure from any source.

From the very beginning, we wanted to design a system small enough in price and size to afford its use at the lowest levels of command. This was another reason for using the modular design. In our opinion, TAC* can be implemented on a cluster of single-board computers. Alternatively, a single minicomputer might suffice for speed and size but would gain nothing from the message-passing independence of the nodes, which with SBC's implies "plug and go" repairs.

The amount of required storage for the database and knowledge base is another matter. A reasonable estimate of required storage is, we feel, about 20-50 megabytes. Obviously, semiconductor RAM would reduce access time, but would increase system cost. A hard disk would bring the

price down somewhat, but access time would suffer. Also, the feasibility of the hard disk to the tactical environment is suspect as they are fairly susceptible to shocks. We feel that magnetic bubble memory devices are, or soon will be, dense, cheap, and fast enough to solve this problem. In addition, it is non-volatile which would hasten recovery from crashes and faults.

The overall concept of a "system of computers" should prove to be more conducive to the tactical environment if, for no other reason, than its portable characteristics.

5. Real-Time Response

Much effort has been devoted to develop a knowledge-based computer system which responds in real-time. One of the better known systems to date is MYCIN (Stanford University, 1977)[Ref 5], which was developed as a diagnostic and therapy consultant for bacterial infections in the blood. Many of the basic concepts from MYCIN are also evident in TAC*, however real-time response as defined for the former was not adequate for our tactical system. The impact of this re-definition of real-time response was the necessity to simplify the system control mechanism and the various interfaces wihtout the loss of system

capabilities. Certainly, recent advances in the speed of medium and small computers should prove to be extremely helpful in a successful implementation. But, more importantly, it will be the proper combination of data structures, links, and the method of treating system rules in the same manner as data which will achieve tactical real-time system response for TAC*.

We believe that this real-time response capability is a most important characteristic of our system, for this alone enables the TAC* system to be a useful tool for the lower level commander. Similar "intelligent" computer systems, like MYCIN, may be considered by some to be more sophisticated and powerful than our particular design. However, it is just this sophistication and power, not to mention sheer size, that makes such system's real-time response inappropriate for the tactical environment. TAC* is intentionally designed without assigning alternative probabilities (or certainty factors), an interactive query capability, and other features used to "prove" to the human that the computer's recommendation is the "right" answer. In the tactical environment, we hold that the decision-maker does not have the time to debate relative probabilities with either another human or a machine. Thus, a tactical real-

time response can be achieved, but at the expense of forfeiting a more sophisticated system.

It should be noted, however, that an interactive capability is recognized as being an important feature. Such a capability may easily be incorporated for use in training situations. As the individual becomes proficient with the system, the logic followed by TAC* to make a particular recommendation could be examined by the trainee. The more recommendations TAC* makes which the individual agrees with, the more that individual will "trust" the system's recommendations in a real-world tactical situation.

The capabilities of a TAC* system can be compared to other systems like MYCIN, NTDS, or WWMCCS. We believe that these systems fall short in adaptability, and more specifically, adaptability in tactical real-time. The adaptability of our design was the prime motivation for its conception, with all other considerations being secondary. In this area, we feel that TAC* promises to be far better than either NTDS or WWMCCS, which are not at all adaptable in the sense we have defined.

6. Generalized Concept of TAC*

Many computer systems have been designed for specific problems. The result of such systems is limited

application areas. While the original concept of TAC* was to provide a tool for the tactical commander, the ultimate design was conceived to be general in nature. Due to the method of treating data, rules, and changes to both identically in our design, the basic system may be used in any number of other application areas. Data and rules need not be restricted to those pertaining to our tactical environment scenario. Whether the data and their corresponding rules apply to warfare planning, medicine, automotive production, inventory control, or air traffic control makes no difference to our system. All types of information are processed in the same manner. Such system capability should prove to be an invaluable asset when one considers diminishing operating budgets combined with increasing personnel costs.

Perhaps more than anything else, TAC* is a point of origin from which many other systems can develop. We contend that TAC* is a notion of how intelligent computer systems should be designed to achieve maximum system adaptability. Actual implementations of TAC* will vary, especially in the internal operations of the DBMS and the KBMS. We believe that the general concept of our design is a significant and unique characteristic of the system. No

system re-configuration is needed to implement totally different types of problem environments if the basic system is properly implemented. All one need do is to identify to TAC* a new Database and new Knowledge Base which are applicable to the desired domain of interest.

C. DISADVANTAGES OF TAC*

The acceptance or non-acceptance of TAC* may be viewed as being dependent upon two major factors: real and perceived system disadvantages. Real disadvantages are more easily isolated and, hopefully, can be negated by system modifications. However, perceived disadvantages are not easily overcome. Human receptiveness will probably be more crucial than technical system limitations.

1. Human Receptiveness

Even though computer technology has been incorporated into washing machines, watches, children's toys, video games, etc. many people still view the computer as a mysterious "black box." Very little resistence to computerization is evident so long as the computer provides entertainment or decreases the human workload. However, the slightest hint that computers might participate in the human decision-making process results in human resistance.

TAC*, being typified as a knowledge-based decision-support system, will probably be regarded by many people as a threat to the "ultimate" human characteristic: the ability to reason. Such people will probably not recognize that TAC* and similar intelligent computer systems are merely attempting to simulate and augment the human decision-making process. Regardless of the purported sophistication of the system, and more specifically the system's software, the computer only does that which the human has programmed it to do. MYCIN is led to a particular diagnosis based upon pre-assigned certainty factors and current test results, both of which are input to the system by qualified medical personnel. Likewise, TAC* makes a "decision" or recommendation based upon the current answer to various "WHAT...IF" questions. The list of possible answers are pre-programmed into the system by humans.

Such a simple and logical explanation may fall on deaf ears, though, as most humans are irrational when they feel threatened. Therefore some, and perhaps the majority, of people may simply refuse to accept TAC* for what it really is: another tool at man's disposal.

2. Complexity of System

Because TAC* has been highly modularized, there will necessarily be a significant amount of data exchanged between modules. Even though much effort was devoted to the simplification of the modules and their interfaces, the complexity of the tactical environment itself imposes a limitation upon system simplification. Although the tasks of the individual modules are quite simple, the resultant overall design is more complex than that which was originally conceived.

Complexity has several disadvantages and TAC* will not be immune to problems. A complex design is more expensive in terms of both dollars and man-hours spent to implement and maintain, increases the possibility of system "bugs", is more difficult for the average person to understand, and therefore, to accept, and normally experiences a higher incidence of "software tampering" than does a less complex design.

People tend to relate complexity to reliability. Due to the stakes involved in the tactical environment, system reliability is a non-negotiable requirement. The perception of design complexity may also create the perception of an unreliable system.

3. System Control

Just as in the human decision-making process, the effectiveness of a computerized decision-support system is directly related to the quality of the information used to make a particular decision. Therefore, some method of system control must be enforced to insure that only authorized rule changes are input, only reliable data updates are made, and all unauthorized access attempts are denied and reported. Such protection and security requirements are not easily accomplished, as evidenced by the vast number of both civilian and military research projects that are on-going in these areas. Further, the reliability of data seems to imply that some type of formal or informal mathematical probability must be used. Anything short of probability is mere speculation.

The result of making an error in the tactical environment could be fatal to the single unit and could have disastrous effects on the national goals of the forces involved. Because it is feasible to configure TAC* in such a way that it could actually control various weapons systems, some form of over-ride (or "fail-safe") mechanism must be incorporated to prevent a potential catastrophe from occuring. In its basic form, TAC* is characterized as just

another tool. But in this advanced form, TAC* is both the tool and the wielder of the tool (subject to human supervision). If it is not properly used, unpredictable action may result.

Total system control is mandatory, but may not be feasible with current technology or human disregard for caution.

D. RECOMMENDATIONS

1. Implementation

Implementation of the TAC* system should be done as a cluster of 16- or 32-bit single-board computers, using magnetic bubble memory as storage for the DB and KB. A language suited to the message-passing environment and which is amenable to bit-checking operations should be used.

2. Extensions

An interactive query system (IQS) similar to that used by MYCIN might be useful (see previous discussion in this chapter). Such a capability could be used by the tactical commander or by a student to extract from the system the path of logic used in reaching a particular decision. This information will be available in the form of Log Recorder entries. An IQS module could access the Log file as its own database, with read-only protection, of

course. Because of the time constraints of the tactical commander, this capability might be limited to training purposes.

Additionally, the log file could be read by an Error Analysis Module which would detect, through comparison between recommendations and actual events, which rules of the knowledge base were inconsistent with actual occurrences. These rules could be flagged to the operator for correction, thus providing a valuable service in maintaining the accuracy of the knowledge base.

A capability for the system to develop rules on its own could also be included. Rule generation could be accomplished by rote learning or by the laws of induction and deduction. This might present a problem to the tactician in that he would not know exactly what rules explicitly exist in the knowledge base, but TAC* could identify these to the TAC as logical extensions to its programmed rules.

Finally, TAC* could be extended to implement the recommendations or the decisions which it makes. TAC* would then be a decision-making system, rather than a decision-support system. Constraints could be placed on this capability, such as positive control or control by negation.

If done, the capability of a TAC* platform to respond quickly to all threats would be greatly enhanced.

3. Warning

When a computer system enters the realm of tactical decision-making, it becomes "responsible" for many human lives. This situation will not be borne any easier as situations become more complex. What is placed in the knowledge base of a TAC* system must accurately reflect the policy of the military and be approved by a competent authority. Some method of testing or verifying the correctness of the system must be considered and used. This must be accomplished before TAC* assumes the role of computerized decision-maker.

E. CONCLUSION

In the final analysis, we believe that we have accomplished that which we set out to accomplish. The top-level, conceptual design of a knowledge-based, intelligent decision-support system has been presented and the bases of our design have been made known. The proposed system, we feel, will be small, flexible, and adaptable. It is our opinion that TAC* will prove helpful in the design of future tactical command and control systems.

Continued work is needed to refine the concept and to test its practicality by various implementation strategies. However, the basis for TAC* must be kept foremost in any attempt at implementation: intelligence and adaptability in real-time.

LIST OF REFERENCES

1. Newell, Allen, and Simon, Herbert A., Human Problem Solving, Prentice-Hall, 1972.

2. Klix, Friedhart, Human and Artificial Intelligence, North-Holland, 1979.

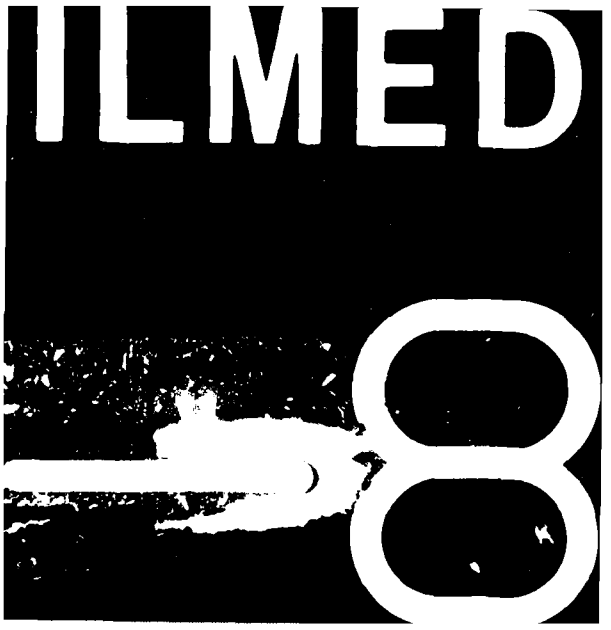
3. Glass, A.L., Holyoak, K.J., and Santa, J.L., Cognition, Addison-Wesley, 1979.

4. Nilsson, Nils J., Principles of Artificial Intelligerce, Tioga, 1980.

5. Davis, R., Buchanon, B., and Shortliffe, Edward, "Production Rules as a Representation for a Knowledge-Based Consultation Program", Artificial Intelligence, Vol. 8, North-Holland, 1977.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 52Bz Computer Science Department Naval Postgraduate School Monterey, California 93940	2
4.	Prof. Douglas Smith, Code 52Sc Computer Science Department Naval Postgraduate School Monterey, California 93940	2
5.	Prof. Lyle Cox, Code 52Cl Computer Science Department Naval Postgraduate School Monterey, California 93940	2
6.	Deputy Under Secretary of the Army for Operations Research Room 2E261, Pentagon Washington, D.C. 20310	2
7.	LCDR William C. Clair 419 Ives Avenue Carneys Point, NJ 08069	2
8.	CPT Ronald F. Danhof 308 Parkview Drive McMinnville, Tennessee 37110	2


ILMED
8